## 1. FoolBox

FoolBox - библиотека для проведения соревновательных (adversarial) атак на нейронные сети

Основные особенности:

- Поддерживает множество классичесих атак: FGSM, PGD, DeepFool, Carlini–Wagner, Boundary attack ...
- Совместим с с PyTorch, TensorFlow, JAX через библиотеку EagerPy
- Позволяет оценивать устойчивость моделей: вычислять вероятность успешной атаки, величину возмущений, сравнивать разные методы защиты
- Архитектура библиотека спроектирована так, что можно атаковать любые модели, работающие с тензорами: не только изображения, но и аудио, тексты, временные ряды.

Основное назначение:

- Используется исследователями для тестирования и сравнения моделей по устойчивости к adversarial-атакам.
- Применяется при разработке новых методов защиты.
- Является стандартным инструментом для проведения экспериментов в области adversarial machine learning.

## 2. WavMark

WavMark — это библиотека для цифрового встраивания водяных знаков (watermarking) в аудиофайлы формата WAV, созданная на базе PyTorch.

Состоит из двух частей:

- Encoder — встраивает водяной знак (обычно битовую последовательность или эмбеддинг) в аудиосигнал.
- Decoder/Detector — извлекает или обнаруживает водяной знак из изменённого/шумного аудио.

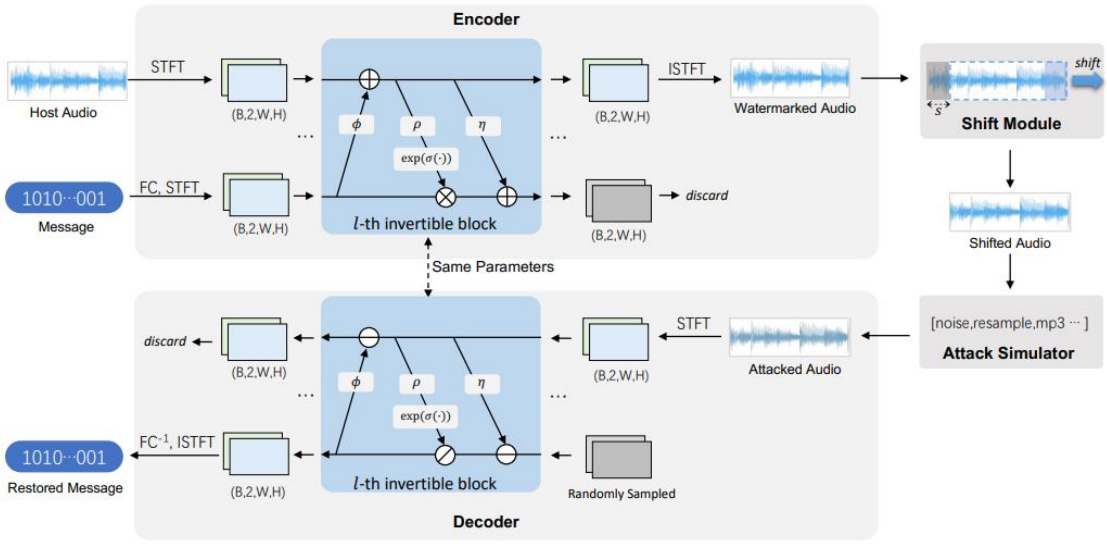


Рисунок 1: Схема эмбеддинга и зашифрования wavmark

Основан на нейронных сетях, которые обучаются так, чтобы:

- Водяной знак был незаметен для слуха (качество звука сохраняется).
- Водяной знак оставался устойчивым к различным искажениям (сжатие, добавление шума, обрезка).
- Работает с сигналами в формате WAV (обычно float или int16).
- Подходит для экспериментов в области робастного аудио-водяного знака и анализа атак на такие системы.

Основное назначение:

- Защита авторских прав и подтверждение подлинности аудиоконтента (музыка, подкасты, синтезированная речь).
- Отслеживание источника данных (например, выявление, был ли файл создан генеративной моделью).
- Исследования в области устойчивости водяных знаков при adversarial-атаках или естественных искажениях сигнала.

# 3. Применимость Foolbox к WavMark

В этом проекте, использовал WavMark для:

- Эмбеддинг — аудио приводится к mono 16 кГц, режется на чанки по 16 000, к каждому применяется model.encode(chunk, payload), затем чанки конкатенируются и сохраняются.
- Декодирование — аудио снова режется на секунды, к батчу секунд применяется model.decode([B,16000]). Это просто проверить что алгоритм нормально работает, не использовать после аттака

У меня есть 5 разных записей в порядке от test_1.wav до test_5.wav, полезные данные, встроенные в них 32-bit payload:

[0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1]

[1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0]

[1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1]

[1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1]

[0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1]

Дальше, с помощью библиотеки FoolBox, сделад соревновательную атаку, основанную на численных пертурбациях, типа  FGSM (Fast Gradient Sign method) , PGD (Projeted Gradient Descent), DeepFool

Обернить декодер WavMark в модель для Foolbox:

- DecodeWrapper: на выходе — 32 вероятности бит; подходит для FGSM/PGD, если рассматривать каждый бит как «мягкий логит».
- MatchLogitWrapper: сжимает 32 бита в 2 класса — match/mismatch (истинные логиты), особенно DeepFool (нужна чёткая разделяющая граница классов).

## 3.1. Fast Gradient Sign method (FGSM)

Для FGSM, можем принимать такую формулу

$$adv = x + \varepsilon * sign\left(\nabla_x J(\theta, x, y)\right)$$

Параметры x – входное аидио, $\nabla_x J(\theta, x, y)$ - производная функции потерь по x (FoolBox сам вычислит), только параметр $\varepsilon$ нужен смотреть

Когда $\varepsilon = 0$, тогда нечего не изменяютя, например выбрем $\varepsilon$ – один из [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0], после декодирования получим

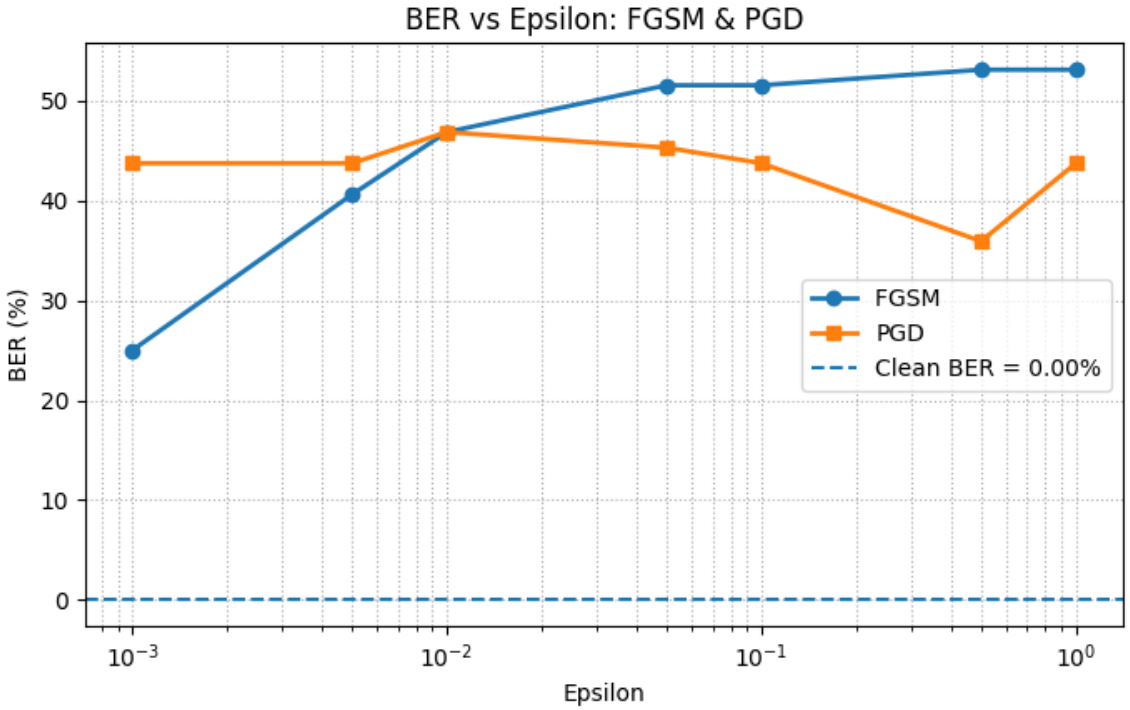


Рисунок 2: Отношение $\varepsilon$ и BER (bit error rate)

## 3.2. Projeted Gradient Descent(PGD)

Метод PGD принимает FGSM несколько раз, более детали смотрим на параметры функции

attack = LinfPGD(steps=10, rel_stepsize=0.5, random_start=True)

steps=10 - Количество итераций PGD. Каждая итерация = один «маленький шаг FGSM» + проекция (project) обратно в шар L

- Больше итераций, атака сильнее, но медленнее.
- Меньше итераций, быстрее, но может быть недостаточно, чтобы «продавить» поверхность потерь.

rel_stepsize=0.5 - Размер шага α: α = rel_stepsize×ε. Рекомендованный диапазон rel_stepsize: 0.3–0.7. Тогда изменим ε можем получим разные α (Рис.2)

random_start=True - Случайная инициализация внутри L шара радиуса ε вокруг $x_0$ перед началом итераций.

- Помогает не «застревать» на границе раздела; обычно повышает успешность атаки.
- Если False, PGD стартует ровно из $x_0$

## 3.3. DeepFool

В методе DeepFool найдут минимальное по L2 возмущение, которое переведёт образ через границу классов.

Алгоритм итеративно линеаризует модель вокруг текущей точки и шагает к ближайшей гиперплоскости раздела.

Для бинарного случая это «коротчайший перпендикуляр» к локальной границе; для многоклассового — выбирается ближайшая из гиперплоскостей «текущий класс ↔ другой класс».

DeepFool - поиск кратчайшего направления L2 к слою «miss/match» и при запуске ε отсутствует, поэтому проецируем на другое $\varepsilon_2$ с помощью project_l2, а затем измеряем BER.

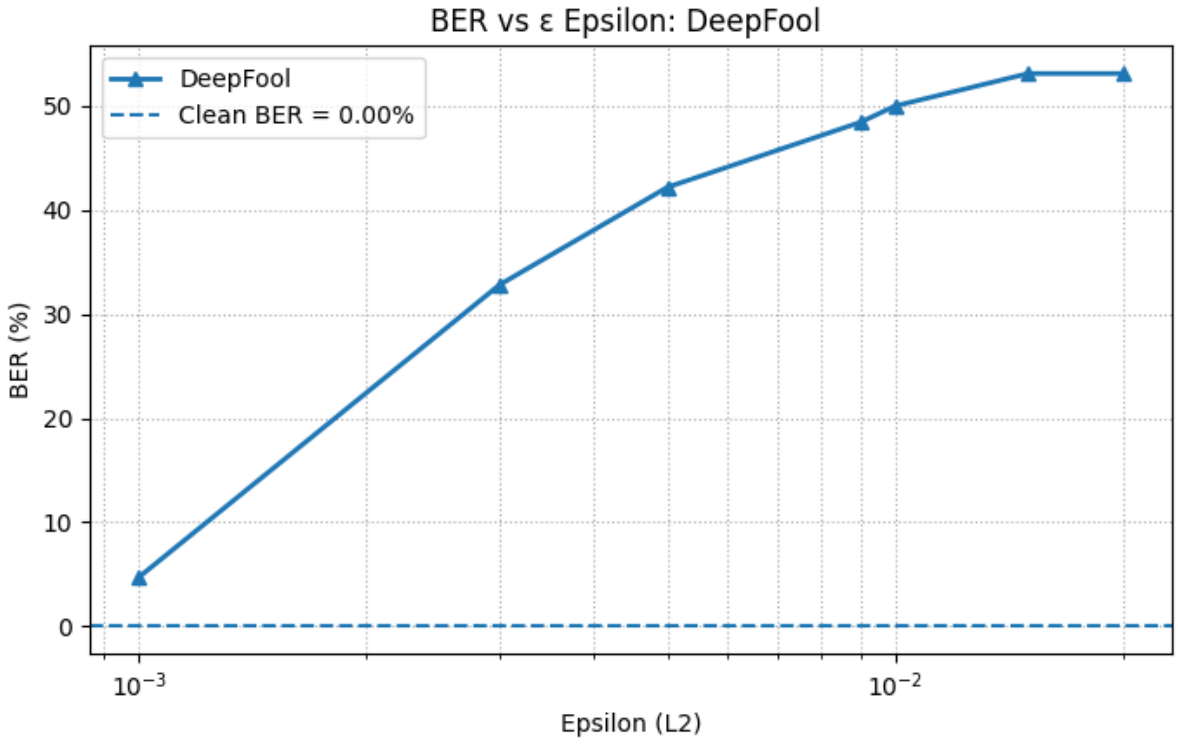


Рисунок 3: Отношение $\varepsilon$ и BER

## 4. Заключение

После анализа 5 аудиозаписей и получения различных результатов (github), я сделал следующие выводы:

По DeepFool: видно, что BER колеблется от 5% до 60%, и кривая имеет тенденцию к росту; чем больше eps2, тем выше BER, но затем наблюдается стремление к некоторому фиксированному предельному значению.

По двум методам FGSM и PGD: хотя BER колеблется от 30% до 60%, графики выглядят несколько хаотично и без чёткой тенденции. Возможно, это связано с тем, что внедряемый payload состоит только из 0 и 1 и недостаточно разнообразен.

С теоретической точки зрения wavmark и foolbox можно применять для внесения помех в передаваемую через аудио информацию.